//Soumya Panigrahi

//DSA September 2023 batch

Q1.

<https://practice.geeksforgeeks.org/problems/search-in-linked-list-1664434326/1?utm_source=geeksforgeeks&utm_medium=ml_article_practice_tab&utm_campaign=article_practice_tab>

/\* Node of a linked list

class Node {

int data;

Node next;

Node(int d) { data = d; next = null; }

}

\*/

class Solution {

static boolean searchKey(int n, Node head, int key) {

// Code here

Node current = head;

while (current != null) {

if (current.data == key) {

return true;

}

current = current.next;

}

return false;

}

}

Q2.

<https://practice.geeksforgeeks.org/problems/node-at-a-given-index-in-linked-list/1?utm_source=geeksforgeeks&utm_medium=ml_article_practice_tab&utm_campaign=article_practice_tab>

class GfG

{

public static int getNth(Node node, int index)

{

//Your code here

if (index < 0 || index >= size()) {

throw new IndexOutOfBoundsException("Index out of bounds");

}

Node current = head;

for (int i = 0; i < index; i++) {

current = current.next;

}

return current.data;

}

}

Q3.

<https://practice.geeksforgeeks.org/problems/count-nodes-of-linked-list/1?utm_source=geeksforgeeks&utm_medium=ml_article_practice_tab&utm_campaign=article_practice_tab>

//{ Driver Code Starts

import java.util.\*;

class Node{

int data;

Node next;

Node(int a){

data = a;

next = null;

}

}

// } Driver Code Ends

/\*Complete the function below\*/

/\*

class Node{

int data;

Node next;

Node(int a){ data = a; next = null; }

}\*/

class Solution

{

//Function to count nodes of a linked list.

public static int getCount(Node head)

{

if(head == null){ return 0; }

int count = 0;

Node current = head;

while(current != null){

count++;

current = current.next;

}

return count;

}

}

//{ Driver Code Starts.

class LinkedList{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int t=sc.nextInt();

while(t-->0)

{

int n=sc.nextInt();

Node head = new Node(sc.nextInt());

Node tail = head;

for(int i=0; i<n-1; i++)

{

tail.next = new Node(sc.nextInt());

tail = tail.next;

}

Solution ob = new Solution();

System.out.println(ob.getCount(head));

}

}

}

// } Driver Code Ends

Q4.

<https://practice.geeksforgeeks.org/problems/insert-in-middle-of-linked-list/1?utm_source=geeksforgeeks&utm_medium=ml_article_practice_tab&utm_campaign=article_practice_tab>

//{ Driver Code Starts

import java.util.\*;

import java.io.\*;

class Node {

int data;

Node next;

public Node(int data){

this.data = data;

this.next = null;

}

}

public class Main {

public static void main(String[] args){

Scanner sc = new Scanner (System.in);

int t = sc.nextInt();

while(t-->0){

int n = sc.nextInt();

Node head = new Node(sc.nextInt());

Node tail = head;

for(int i=0; i<n-1; i++){

tail.next = new Node(sc.nextInt());

tail = tail.next;

}

int key = sc.nextInt();

//display(head);

Solution obj = new Solution();

obj.insertInMid(head, key);

display(head);

System.out.println();

}

}

public static void display(Node head){

Node curr = head;

while(curr != null){

System.out.print(curr.data+" ");

curr = curr.next;

}

}

}

// } Driver Code Ends

/\*

Structure of node class is:

class Node {

int data;

Node next;

public Node(int data){

this.data = data;

this.next = null;

}

}

\*/

class Solution {

public Node insertInMid(Node head, int data){

//Insert code here, return the head of modified linked list

// int cnt=0;

// int target = 0;

// Node curr = head;

// for(; curr != null; ++cnt, curr = curr.next);

// target = cnt / 2 - 1;

// for(curr = head; target>=0; curr = curr.next, --target);

// return curr;

Node temp = new Node(data);

if(head==null)

{

return head;

}

Node slow=head;

Node fast = head;

while(fast.next!=null&&fast.next.next!=null)

{

fast = fast.next.next;

slow=slow.next;

}

temp.next = slow.next;

slow.next = temp;

return head;

}

}

Q5.

<https://practice.geeksforgeeks.org/problems/find-the-sum-of-last-n-nodes-of-the-linked-list/1?utm_source=geeksforgeeks&utm_medium=ml_article_practice_tab&utm_campaign=article_practice_tab>

class GfG

{

Node head;

void insert(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

} else {

Node temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = newNode;

}

}

int getNodeData(int index) {

Node temp = head;

for (int i = 1; i < index; i++) {

temp = temp.next;

}

return temp.data;

}

}

Q6.

<https://leetcode.com/problems/merge-two-sorted-lists/>

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* int val;

\* ListNode next;

\* ListNode() {}

\* ListNode(int val) { this.val = val; }

\* ListNode(int val, ListNode next) { this.val = val; this.next = next; }

\* }

\*/

class Solution {

public ListNode mergeTwoLists(ListNode list1, ListNode list2) {

ListNode p1 = list1, p2 = list2;

ListNode gift = new ListNode(0);

ListNode p = gift;

while(p1 != null && p2 != null){

if(p1.val <= p2.val){ p.next = p1; p1 = p1.next; }

else { p.next = p2; p2 = p2.next; }

p = p.next;

}

if(p1 != null) { p.next = p1; }

if(p2 != null) { p.next = p2; }

return gift.next;

}

}

Q7.

<https://leetcode.com/problems/sort-list/>

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* int val;

\* ListNode next;

\* ListNode() {}

\* ListNode(int val) { this.val = val; }

\* ListNode(int val, ListNode next) { this.val = val; this.next = next; }

\* }

\*/

class Solution {

public ListNode merge(ListNode list1, ListNode list2) {

ListNode p1 = list1, p2 = list2;

ListNode gift = new ListNode(0);

ListNode p = gift;

while(p1 != null && p2 != null){

if(p1.val <= p2.val){ p.next = p1; p1 = p1.next; }

else { p.next = p2; p2 = p2.next; }

p = p.next;

}

if(p1 != null) { p.next = p1; }

if(p2 != null) { p.next = p2; }

return gift.next;

}

public ListNode sortList(ListNode head) {

if(!head || !head.next) return head;

ListNode speedy = head, speedyx1oo = head.next;

while(speedyx1oo && speedyx1oo.next){

speedy = speedy.next;

speedyx1oo = speedyx1oo.next.next;

}

speedyx1oo = speedy.next;

speedy.next = null;

head = sortList(head);

speedyx1oo = sortList(speedyx1oo);

return merge(head, speedyx1oo);

}

}

Q8.

<https://leetcode.com/problems/reverse-linked-list/>

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* int val;

\* ListNode next;

\* ListNode() {}

\* ListNode(int val) { this.val = val; }

\* ListNode(int val, ListNode next) { this.val = val; this.next = next; }

\* }

\*/

class Solution {

public ListNode reverseList(ListNode head) {

if(head == null || head.next == null) {

return head;

}

ListNode prevNode = head;

ListNode currNode = head.next;

while(currNode != null) {

ListNode nextNode = currNode.next;

currNode.next = prevNode;

prevNode = currNode;

currNode = nextNode;

}

head.next = null;

head = prevNode;

return head;

}

}

Q9.

<https://leetcode.com/problems/palindrome-linked-list/>

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* int val;

\* ListNode next;

\* ListNode() {}

\* ListNode(int val) { this.val = val; }

\* ListNode(int val, ListNode next) { this.val = val; this.next = next; }

\* }

\*/

class Solution {

public boolean isPalindrome(ListNode head) {

if(head == null){return false;}

if(head.next == null){return true;}

ListNode mid = midnode(head);

ListNode l2 = mid.next;

mid.next = null;

l2 = reverseList(l2);

return compare(head, l2);

}

public boolean compare(ListNode l1, ListNode l2){

while(l1 != null && l2 != null){

if(l1.val != l2.val){return false;}

l1 = l1.next;

l2 = l2.next;

}

return true;

}

public ListNode reverseList(ListNode head){

ListNode prev = null, curr = head, temp;

while(curr != null){

temp = curr.next;

curr.next = prev;

prev = curr; curr = temp;

}

return prev;

}

public ListNode midnode(ListNode head){

ListNode fast = head, slow = head;

while(fast.next != null && fast.next.next != null){

fast = fast.next.next;

slow = slow.next;

}

return slow;

}

}

Q10.

<https://practice.geeksforgeeks.org/problems/detect-loop-in-linked-list/1?utm_source=gfg&utm_medium=article&utm_campaign=bottom_sticky_on_article>

class Solution {

//Function to check if the linked list has a loop.

public static boolean detectLoop(Node head){

// Add code here

Node speedy = head;

Node speedyxhundred = head;

while(speedyxhundred != null && speedyxhundred.next != null){

speedy = speedy.next;

speedyxhundred = speedyxhundred.next.next;

if(speedyxhundred == speedy){

return true;

}

}

return false;

}

}

Q11.

<https://leetcode.com/problems/reverse-nodes-in-k-group/>

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* int val;

\* ListNode next;

\* ListNode() {}

\* ListNode(int val) { this.val = val; }

\* ListNode(int val, ListNode next) { this.val = val; this.next = next; }

\* }

\*/

class Solution {

public ListNode reverseKGroup(ListNode head, int k) {

if(head == null || k==1){return head;}

//ListNode \*dummy = malloc(sizeof( struct ListNode\*));

ListNode dummy = new ListNode(0);

dummy.next = head;

ListNode curr = dummy, prev=dummy, nest=dummy;

int count = 0;

while (curr.next != null)

{

curr = curr.next;

count++;

}

while (count >= k)

{

curr = prev.next;

nest = curr.next;

for (int i = 01; i < k; i++)

{

curr.next = nest.next;

nest.next = prev.next;

prev.next = nest;

nest = curr.next;

}

prev = curr;

count -= k;

}

return dummy.next;

}

}

Q12.

<https://leetcode.com/problems/middle-of-the-linked-list/>

//code 1 successfully running in compiler

class Solution{

public ListNode middleNode(ListNode head){

int cnt=0;

int target = 0;

ListNode curr = head;

for(; curr != null; ++cnt, curr = curr.next);

target = cnt / 2 - 1;

for(curr = head; target>=0; curr = curr.next, target--);

return curr;

}

}

// code 2 successfully have run

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* int val;

\* ListNode next;

\* ListNode() {}

\* ListNode(int val) { this.val = val; }

\* ListNode(int val, ListNode next) { this.val = val; this.next = next; }

\* }

\*/

class Solution {

public ListNode middleNode(ListNode head) {

if(head == null)return head;

ListNode speedy = head, speedyx1oo = head;

while(speedyx1oo != null && speedyx1oo.next != null) {

speedy = speedy.next;

speedyx1oo = speedyx1oo.next.next;

}

return speedy;

}

}

Q13.

<https://leetcode.com/problems/merge-two-sorted-lists/>   (do recursively)

class Solution {

public ListNode mergeTwoLists(ListNode list1, ListNode list2) {

// ListNode p1 = list1, p2 = list2;

if(list1 == null) return list2;

if(list2 == null)return list1;

if(list1.val <= list2.val)

{list1.next = mergeTwoLists(list1.next, list2);

return list1;}

else{list2.next = mergeTwoLists(list1, list2.next);

return list2;}

//return list2;

}

}

Q14.

<https://leetcode.com/problems/remove-duplicates-from-sorted-list/>

class Solution {

public ListNode deleteDuplicates(ListNode head) {

if(head==null)return head;

ListNode newn =new ListNode(head.val);

ListNode res=newn;

ListNode curr=head;

while(curr!=null)

{

if(newn.val!=curr.val)

{

newn.next=new ListNode(curr.val);

newn=newn.next;

}

curr=curr.next;

}

return res;

}

}

Q15.

<https://leetcode.com/problems/swap-nodes-in-pairs/>

class Solution {

public ListNode swapPairs(ListNode head) {

ListNode slow = head, fast = head, temp = head;

int k = 2;

for(int i=0; i<k; i++){

temp = fast; fast = fast.next;

//temp = fast;

}

while(fast.next != null){

fast = fast.next;

slow = slow.next;

}

int temp2 = temp.val;

temp.val = slow.val;

slow.val = temp2;

return head;

}

}

Q16.

<https://leetcode.com/problems/partition-list/>

class Solution {

public ListNode partition(ListNode head, int x) {

if(head == null)return head;

ListNode h1 = new ListNode(0);

ListNode h2 = new ListNode(0);

h1.next = head;

ListNode p = head, prev = h1, prev2 = h2;

while(p != null){

if(p.val < x){p=p.next; prev = prev.next;}

else{ prev2.next = p; prev.next = p.next; p = prev.next; prev2 = prev2.next; }

}

prev2.next = null;

prev.next = h2.next;

return h1.next;

}

}